

GNAP
Internet-Draft
Intended status: Standards Track
Expires: 13 January 2022

J. Richer, Ed.
Bespoke Engineering
A. Parecki
Okta
F. Imbault
acert.io
12 July 2021

Grant Negotiation and Authorization Protocol Resource Server Connections [draft-ietf-gnap-resource-servers-01](#)

Abstract

GNAP defines a mechanism for delegating authorization to a piece of software, and conveying that delegation to the software. This extension defines methods for resource servers (RS) to communicate with authorization servers (AS) in an interoperable fashion.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 13 January 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the [Trust Legal Provisions](#) and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	2
1.1.	Terminology	3
2.	Access Token Formats	3
3.	Resource-Server-Facing API	4
3.1.	RS-facing AS Discovery	4
3.2.	Protecting RS requests to the AS	5
3.3.	Token Introspection	6
3.4.	Registering a Resource Set	9
4.	Deriving a downstream token	11
5.	Acknowledgements	13
6.	IANA Considerations	13
6.1.	GNAP Token Format Registry	13
6.1.1.	Registry Template	13
6.1.2.	Initial Registry Contents	13
7.	Security Considerations	14
8.	Privacy Considerations	14
9.	Normative References	14
Appendix A.	Document History	15
	Authors' Addresses	15

[1.](#) Introduction

The core GNAP protocol does not define any one specific mechanism for the resource server (RS) to communicate with the authorization server (AS), allowing the connection between these components to be solved orthogonally to the core protocol's concerns. For example, the RS and AS roles could be fulfilled by the same piece of software with common storage, obviating the need for any connecting protocol. However, it is often desirable to have the RS and AS communicate at runtime for a variety of purposes, including allowing the RS to validate and understand the rights and privileges associated with a grant of access represented by an access token issued by the AS, or negotiating the capabilities of either party. These types of connections are particularly useful for connecting an AS and RS from different vendors, allowing interoperable distributed deployments of GNAP-protected systems.

This specification defines several means for a RS and AS to communicate these aspects with each other, including structured access tokens and RS-facing APIs. This specification also discusses methods for an RS to derive a downstream token for calling another chained RS.

The means of the authorization server issuing the access token to the client instance and the means of the client instance presenting the access token to the resource server are the subject of the GNAF core protocol specification [[I-D.ietf-gnaf-core-protocol](#)].

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [BCP 14](#) [[RFC2119](#)] [[RFC8174](#)] when, and only when, they appear in all capitals, as shown here.

This document contains non-normative examples of partial and complete HTTP messages, JSON structures, URLs, query components, keys, and other elements. Some examples use a single trailing backslash ' ' to indicate line wrapping for long values, as per [[RFC8792](#)]. The "\" character and leading spaces on wrapped lines are not part of the value.

Terminology specific to GNAF is defined in the terminology section of the core specification [[I-D.ietf-gnaf-core-protocol](#)], and provides definitions for the protocol roles: Authorization Server (AS), Client, Resource Server (RS), Resource Owner (RO), End-user; as well as the protocol elements: Attribute, Access Token, Grant, Privilege, Protected Resource, Right, Subject, Subject Information. The same definitions are used in this document.

2. Access Token Formats

When the AS issues an access token for use at an RS, the RS needs to have some means of understanding what the access token is for in order to determine how to respond to the request. The core GNAF protocol makes no assumptions or demands on the format or contents of the access token, and in fact the token format and contents are opaque to the client instance. However, such token formats can be the topic of agreements between the AS and RS.

Self-contained structured token formats allow for the conveyance of information between the AS and RS without requiring the RS to call the AS at runtime as described in [Section 3.3](#). Structured tokens can also be used in combination with introspection, allowing the token itself to carry one class of information and the introspection response to carry another.

Some token formats, such as Macaroons and Biscuits, allow for the RS to derive sub-tokens without having to call the AS as described in [Section 4](#).

The supported token formats can be communicated dynamically at runtime between the AS and RS in several places.

- * The AS can declare its supported token formats as part of RS-facing discovery [Section 3.1](#)
- * The RS can require a specific token format be used to access a registered resource set [Section 3.4](#)
- * The AS can return the token's format in an introspection response [Section 3.3](#)

In all places where the token format is listed explicitly, it MUST be one of the registered values in the GNAP Token Formats Registry [Section 6.1](#).

[3.](#) Resource-Server-Facing API

To facilitate runtime and dynamic connections, the AS can offer an RS-Facing API consisting of one or more of the following optional pieces.

- * Discovery
- * Introspection
- * Token chaining
- * Resource reference registration

[3.1.](#) RS-facing AS Discovery

A GNAP AS offering RS-facing services can publish its features on a well-known discovery document using the URL ".well-known/gnap-as-rs" appended to the grant request endpoint URL.

The discovery response is a JSON document [[RFC8259](#)] consisting of a single JSON object with the following fields:

`introspection_endpoint` (string): OPTIONAL. The URL of the endpoint offering introspection. The location MUST be a URL [[RFC3986](#)] with a scheme component that MUST be https, a host component, and optionally, port, path and query components and no fragment components. [Section 3.3](#)

`token_formats_supported` (array of strings): A list of token formats supported by this AS. The values in this list MUST be registered in the GNAAP Token Format Registry. [Section 6.1](#)

`resource_registration_endpoint` (string): The URL of the endpoint offering resource registration. The location MUST be a URL [[RFC3986](#)] with a scheme component that MUST be https, a host component, and optionally, port, path and query components and no fragment components. [Section 3.4](#)

`grant_request_endpoint` (string): REQUIRED. The location of the AS's grant request endpoint, used by the RS to derive downstream access tokens. The location MUST be a URL [[RFC3986](#)] with a scheme component that MUST be https, a host component, and optionally, port, path and query components and no fragment components. This URL MUST be the same URL used by client instances in support of GNAAP requests. [Section 4](#)

`key_proofs_supported` (array of strings) OPTIONAL. A list of the AS's supported key proofing mechanisms. The values of this list correspond to possible values of the "proof" field of the key section of the request.

[3.2.](#) Protecting RS requests to the AS

Unless otherwise specified, the RS MUST protect its calls to the AS using any of the signature methods defined by GNAAP. This signing method MUST cover all of the appropriate portions of the HTTP request message, including any body elements, tokens, or headers required for functionality.

The RS MAY present its keys by reference or by value in a similar fashion to a client instance calling the AS in the core protocol of GNAAP, described in [[I-D.ietf-gnap-core-protocol](#)]. In the protocols defined here, this takes the form of the resource server identifying itself using a "key" field or by passing an instance identifier directly.

```

"resource_server": {
  "key": {
    "proof": "httpsig",
    "jwk": {
      "kty": "EC",
      "crv": "secp256k1",
      "kid": "2021-07-06T20:22:03Z",
      "x": "-J90JIZj4nmopZbQN7T8xv3sbeS5-f_vBNSy_EHnBZc",
      "y": "sjrS51pLtu3P4LUTVvyAIXRfDV_be2RYpI5_f-Yjivw"
    }
  }
}

```

or by reference:

```
"resource_server": "7C7C4AZ9KHRS6X63AJA0"
```

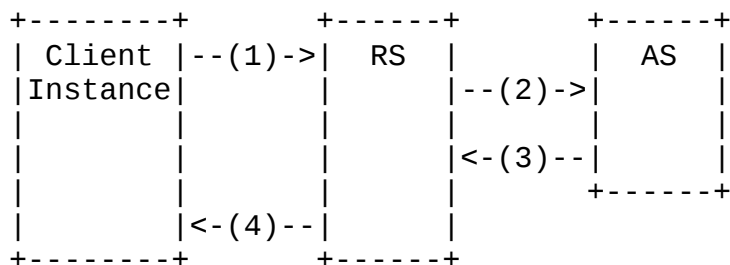
The AS MAY require an RS to pre-register its keys or could allow calls from arbitrary keys in a trust-on-first-use model.

3.3. Token Introspection

The AS issues access tokens representing a set of delegated access rights to be used at one or more RSs. The AS can offer an introspection service to allow an RS to validate that a given access token:

- * has been issued by the AS
- * has not expired
- * has not been revoked
- * is appropriate for the RS identified in the call

When the RS receives an access token, it can call the introspection endpoint at the AS to get token information. [[See issue #115 (<https://github.com/ietf-wg-gnap/gnap-core-protocol/issues/115>)]]



1. The client instance calls the RS with its access token.
2. The RS introspects the access token value at the AS. The RS signs the request with its own key (not the client instance's key or the token's key).
3. The AS validates the access token value and the Resource Server's request and returns the introspection response for the token.
4. The RS fulfills the request from the client instance.

The RS signs the request with its own key and sends the access token as the body of the request.

`access_token (string): REQUIRED.` The access token value presented to the RS by the client instance.

`proof (string): RECOMMENDED.` The proofing method used by the client instance to bind the token to the RS request.

`resource_server (string or object): REQUIRED.` The identification used to authenticate the resource server making this call, either by value or by reference as described in [Section 3.2](#).

`access (array of strings/objects): OPTIONAL.` The minimum access rights required to fulfill the request. This MUST be in the format described in the Resource Access Rights section of [\[I-D.ietf-gnap-core-protocol\]](#).

```
POST /introspect HTTP/1.1
Host: server.example.com
Content-Type: application/json
Signature-Input: sig1=...
Signature: sig1=...
Digest: sha256=...
```

```
{
  "access_token": "OS9M2PMHKUR64TB8N6BW7OZB8CDFONP219RP1LT0",
  "proof": "httpsig",
  "resource_server": "7C7C4AZ9KHRS6X63AJA0"
}
```

The AS MUST validate the access token value and determine if the token is active. An active access token is defined as a token that

- * was issued by the processing AS,
- * has not been revoked,

- * has not expired, and
- * is appropriate for presentation at the identified RS.

The AS responds with a data structure describing the token's current state and any information the RS would need to validate the token's presentation, such as its intended proofing mechanism and key material.

active (boolean): REQUIRED. If "true", the access token presented is active, as defined above. If any of the criteria for an active token are not true, or if the AS is unable to make a determination (such as the token is not found), the value is set to "false" and other fields are omitted.

If the access token is active, additional fields from the single access token response structure defined in [\[I-D.ietf-gnap-core-protocol\]](#) are included. In particular, these include the following:

access (array of strings/objects): REQUIRED. The access rights associated with this access token. This MUST be in the format described in the Resource Access Rights section of [\[I-D.ietf-gnap-core-protocol\]](#). This array MAY be filtered or otherwise limited for consumption by the identified RS, including being an empty array.

key (object/string): REQUIRED if the token is bound. The key bound to the access token, to allow the RS to validate the signature of the request from the client instance. If the access token is a bearer token, this MUST NOT be included.

flags (array of strings): OPTIONAL. The set of flags associated with the access token.

The response MAY include any additional fields defined in an access token response and MUST NOT include the access token "value" itself.


```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "active": true,
  "access": [
    "dolphin-metadata", "some other thing"
  ],
  "key": {
    "proof": "httpsig",
    "jwk": {
      "kty": "RSA",
      "e": "AQAB",
      "kid": "xyz-1",
      "alg": "RS256",
      "n": "kOB5rR4Jv0GMeL...."
    }
  }
}
```

3.4. Registering a Resource Set

If the RS needs to, it can post a set of resources as described in the Resource Access Rights section of [[I-D.ietf-gnap-core-protocol](#)] to the AS's resource registration endpoint along with information about what the RS will need to validate the request.

access (array of objects/strings): REQUIRED. The list of access rights associated with the request in the format described in the "Resource Access Rights" section of [[I-D.ietf-gnap-core-protocol](#)].

resource_server (string or object): REQUIRED. The identification used to authenticate the resource server making this call, either by value or by reference as described in [Section 3.2](#).

token_format_required (string): OPTIONAL. The token format required to access the identified resource. If the field is omitted, the token format is at the discretion of the AS. If the AS does not support the requested token format, the AS MUST return an error to the RS.

token_introspection_required (boolean): OPTIONAL. If present and set to "true", the RS expects to make a token introspection request as described in [Section 3.3](#). If absent or set to "false", the RS does not anticipate needing to make an introspection request for tokens relating to this resource set.

The RS MUST identify itself with its own key and sign the request.

```
POST /resource HTTP/1.1
Host: server.example.com
Content-Type: application/json
Signature-Input: sig1=...
Signature: sig1=...
Digest: ...
```

```
{
  "access": [
    {
      "actions": [
        "read",
        "write",
        "dolphin"
      ],
      "locations": [
        "https://server.example.net/",
        "https://resource.local/other"
      ],
      "datatypes": [
        "metadata",
        "images"
      ]
    },
    "dolphin-metadata"
  ],
  "resource_server": "7C7C4AZ9KHRS6X63AJA0"
}
```

The AS responds with a reference appropriate to represent the resources list that the RS presented in its request as well as any additional information the RS might need in future requests.

resource_reference (string): REQUIRED. A single string representing the list of resources registered in the request. The RS MAY make this handle available to a client instance as part of a discovery response as described in [[I-D.ietf-gnap-core-protocol](#)] or as documentation to client software developers.

instance_id (string): OPTIONAL. An instance identifier that the RS can use to refer to itself in future calls to the AS, in lieu of sending its key by value.

introspection_endpoint (string): OPTIONAL. The introspection

endpoint of this AS, used to allow the RS to perform token introspection. [Section 3.3](#)

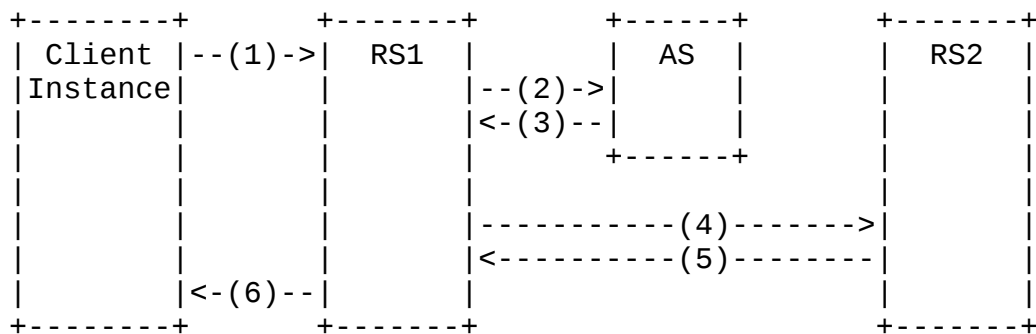
```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "resource_reference": "FWWIKYBQ6U56NL1"
}
```

4. Deriving a downstream token

Some architectures require an RS to act as a client instance and use a derived access token for a secondary RS. Since the RS is not the same entity that made the initial grant request, the RS is not capable of referencing or modifying the existing grant. As such, the RS needs to request or generate a new token access token for its use at the secondary RS. This internal secondary token is issued in the context of the incoming access token.

While it is possible to use a token format ([Section 2](#)) that allows for the RS to generate its own secondary token, the AS can allow the RS to request this secondary access token using the same process used by the original client instance to request the primary access token. Since the RS is acting as its own client instance from the perspective of GNAP, this process uses the same grant endpoint, request structure, and response structure as a client instance's request.



1. The client instance calls RS1 with an access token.
2. RS1 presents that token to the AS to get a derived token for use at RS2. RS1 indicates that it has no ability to interact with the RO. Note that RS1 signs its request with its own key, not the token's key or the client instance's key.

3. The AS returns a derived token to RS1 for use at RS2.
4. RS1 calls RS2 with the token from (3).
5. RS2 fulfills the call from RS1.
6. RS1 fulfills the call from the original client instance.

If the RS needs to derive a token from one presented to it, it can request one from the AS by making a token request as described in [\[I-D.ietf-gnap-core-protocol\]](#) and presenting the existing access token's value in the "existing_access_token" field.

Since the RS is acting as a client instance, the RS MUST identify itself with its own key in the "client" field and sign the request just as any client instance would, as described in [Section 3.2](#).

```
POST /tx HTTP/1.1
Host: server.example.com
Content-Type: application/json
Detached-JWS: ejy0...
```

```
{
  "access_token": {
    "access": [
      {
        "actions": [
          "read",
          "write",
          "dolphin"
        ],
        "locations": [
          "https://server.example.net/",
          "https://resource.local/other"
        ],
        "datatypes": [
          "metadata",
          "images"
        ]
      },
      "dolphin-metadata"
    ],
  },
  "client": "7C7C4AZ9KHRS6X63AJA0",
  "existing_access_token": "OS9M2PMHKUR64TB8N6BW7OZB8CDFONP219RP1LT0"
}
```

The AS responds with a token for the downstream RS2 as described in [\[I-D.ietf-gnap-core-protocol\]](#). The downstream RS2 could repeat this process as necessary for calling further RS's.

5. Acknowledgements

(TODO: the ACK section should probably be split between the documents)

6. IANA Considerations

[[TBD: There are a lot of items in the document that are expandable through the use of value registries.]]

6.1. GNAP Token Format Registry

This specification establishes the GNAP Token Format Registry to define token formats.

6.1.1. Registry Template

6.1.2. Initial Registry Contents

The table below contains the initial contents of the GNAP Token Format Registry.

Name	Status	Description	Reference
jwt-signed	Active	JSON Web Token, signed with JWS	[RFC7519]
jwt-encrypted	Active	JSON Web Token, encrypted with JWE	[RFC7519]
macaroon	Active	Macaroon	
biscuit	Active	Biscuit	
zcap	Active	ZCAP	

Table 1: Initial contents of the GNAP Token Format Registry.

7. Security Considerations

[[TBD: There are a lot of security considerations to add.]]

All requests have to be over TLS or equivalent as per [BCP195]. Many handles act as shared secrets, though they can be combined with a requirement to provide proof of a key as well.

8. Privacy Considerations

[[TBD: There are a lot of privacy considerations to add.]]

When introspection is used, the AS is made aware of a particular token being used at a particular AS, and the AS would not otherwise have insight into this.

When the client instance receives information about the protecting AS from an RS, this can be used to derive information about the resources being protected without releasing the resources themselves.

9. Normative References

[BCP195] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", May 2015, <<https://www.rfc-editor.org/info/bcp195>>.

[I-D.ietf-gnap-core-protocol] Richer, J., Parecki, A., and F. Imbault, "Grant Negotiation and Authorization Protocol", Work in Progress, Internet-Draft, [draft-ietf-gnap-core-protocol-05](https://www.ietf.org/archive/id/draft-ietf-gnap-core-protocol-05), 28 April 2021, <<https://www.ietf.org/archive/id/draft-ietf-gnap-core-protocol-05.txt>>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, [RFC 3986](#), DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", [RFC 7519](#), DOI 10.17487/RFC7519, May 2015, <<https://www.rfc-editor.org/info/rfc7519>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in [RFC 2119](#) Key Words", [BCP 14](#), [RFC 8174](#), DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, [RFC 8259](#), DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", [RFC 8792](#), DOI 10.17487/RFC8792, June 2020, <<https://www.rfc-editor.org/info/rfc8792>>.

[Appendix A](#). Document History

- * -01
 - Better described RS authentication.
 - Added access token format registry.
 - Filled out introspection protocol.
 - Filled out resource registration protocol.
 - Expanded RS-facing discovery mechanisms.
 - Moved client-facing RS response back to GNAP core document.
- * -00
 - Extracted resource server section.

Authors' Addresses

Justin Richer (editor)
Bespoke Engineering

Email: ietf@justin.richer.org
URI: <https://bspk.io/>

Aaron Parecki
Okta

Email: aaron@parecki.com
URI: <https://aaronparecki.com>

Fabien Imbault
acert.io

Email: fabien.imbault@acert.io

URI: <https://acert.io/>